COMPUTER VISION 2024

LAB 6

TABLE OF CONTENTS

OBJECTIVE

- Image Matching using SIFT
- Review RANSAC algorithm
- Homography Matrix – Brief Overview
- Apply SIFT and RANSAC in image stitching

SIFT

Steps

Here is a recap from the lecture on the main steps of SIFT:

1. Constructing a scale space.
2. Laplacian of Gaussian approximation (DOG).
3. Key-point localization.
4. Eliminate edges and low contrast regions.
5. Orientation Assignment.
6. Key-point description.

Built-in Code:

Initially, create the sift object to be used later for detection/description the key-points of theimage:

```
sift = cv2.xfeatures2d.SIFT_create()
```

Then pass a grayscale image and detect the key-points

```
kp = sift.detect(gray,None)
```

or you can detect and describe these key-points as well (useful for matching):

```
kp, dest = sift.detectAndCompute(gray,None)
```

MATCHING

To match features resultant from 2 images, we need them as descriptive as possible. For the sift, we need to compute the descriptors using the `detectAndCompute` method. Then the matches are calculated using brute force algorithm:

```
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
```

Once it is created, two important methods that can be used to get matches are *BFMatcher.match()* and *BFMatcher.knnMatch()*.

First one returns **the best match per query descriptor**.

```
matches = bf.match(des1,des2)
```

The result of the previous line returns a list of DMatch objects. This DMatch object has the following attributes:

- DMatch.distance - Distance between descriptors. The lower, the better it is.
- DMatch.trainIdx - Index of the descriptor in train descriptors
- DMatch.queryIdx - Index of the descriptor in query descriptors
- DMatch.imgIdx - Index of the train image.

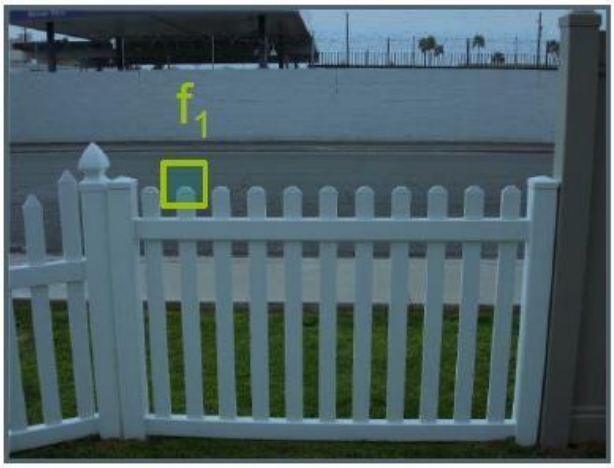
The second method knnMatch returns **k best matches per query descriptor** where k is specified by the user.

```
matches = bf.knnMatch(des1,des2,2)
```

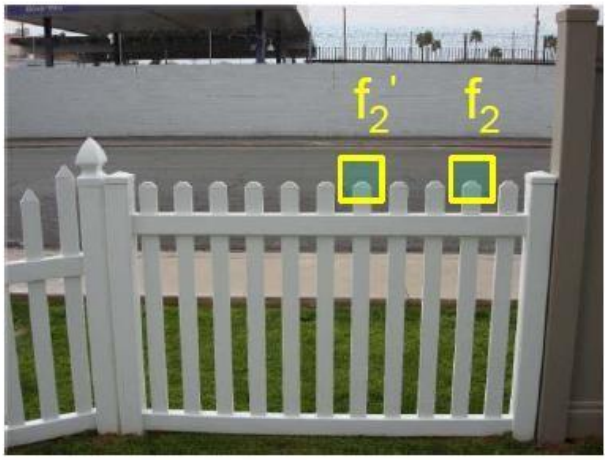
The result of the previous line returns a list of lists of DMatch objects.
Example: [[Dmatch1QueryDes1, Dmatch2QueryDes1], [Dmatch1QueryDes2, Dmatch2QueryDes2], [Dmatch1QueryDes3, Dmatch2QueryDes3], ..]

FEATURE MATCHING - DAVID LOWE'S RATIO

- To find if 2 matching features are True positive or False positive, use Lowe's ratio test:
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2



I1

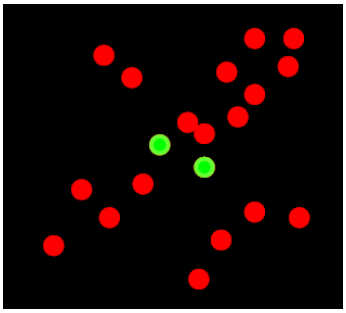


I2

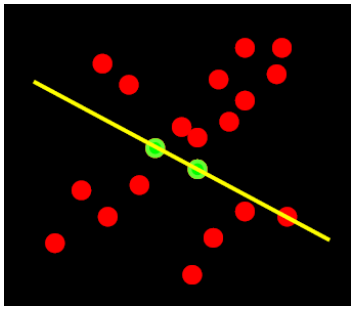
$$\text{TEST: } \frac{SSD(f1,f2)}{SSD(f1,f2')} < \text{ratio}$$

RANSAC ALGORITHM

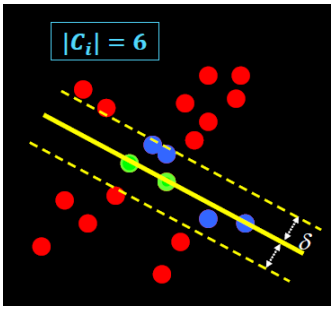
1. Select randomly the minimum number of points required to determine the model parameters.



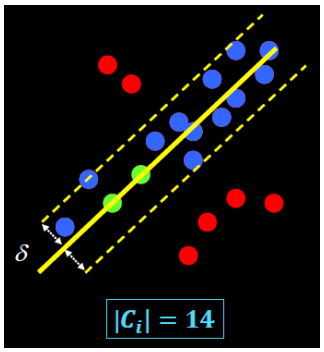
2. Solve for the parameters of the model.



3. Determine how many points from the set of all points fit with a predefined tolerance.



4. If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold τ , re-estimate the model parameters using all the identified inliers and terminate.
5. Otherwise, repeat steps 1 through 4 (maximum of N times)

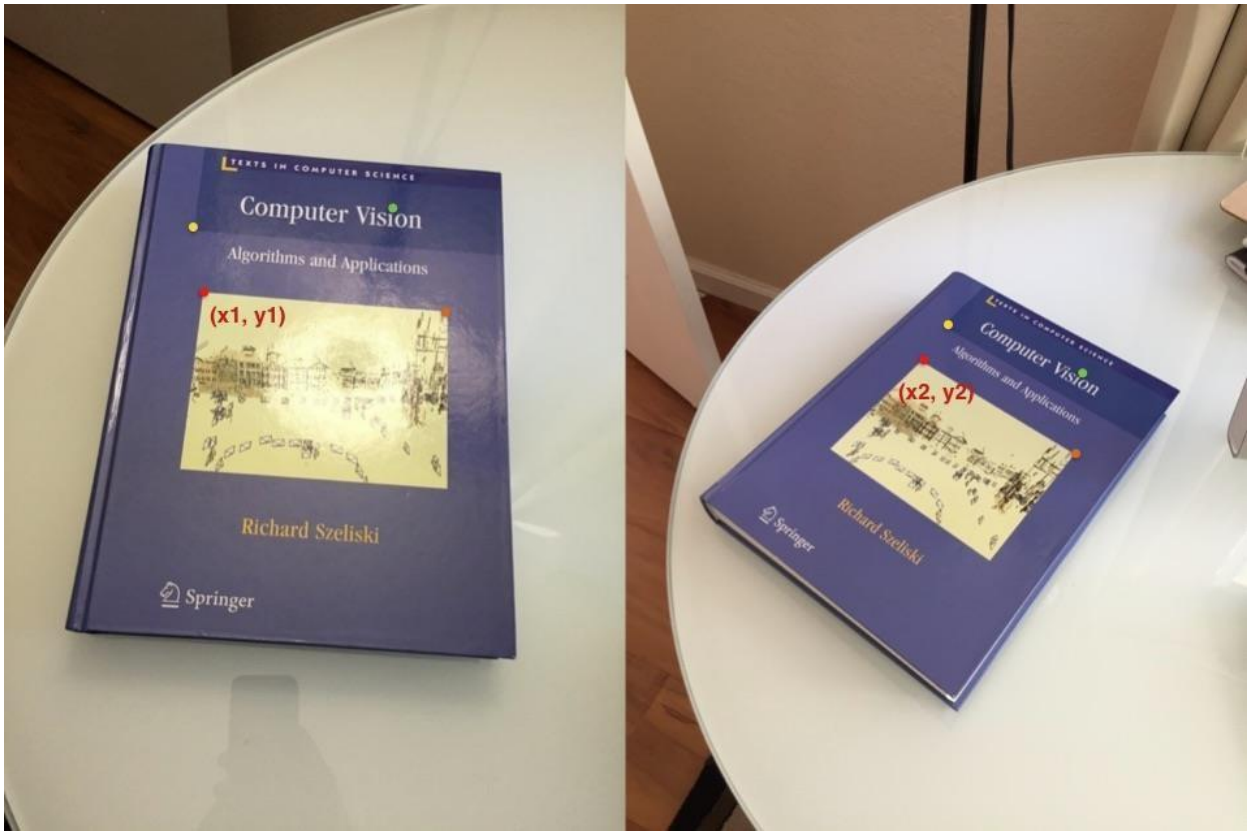


The generic RANSAC algorithm works as follows

```
Given:
    data – A set of observations.
    model – A model to explain observed data points.
    n – Minimum number of data points required to estimate model parameters.
    k – Maximum number of iterations allowed in the algorithm.
    t – Threshold value to determine data points that are fit well by model.
    d – Number of close data points required to assert that a model fits well to data.

Return:
    bestFit – model parameters which best fit the data (or nul if no good model is found)

iterations = 0
bestFit = nul
bestErr = something really large

while iterations < k do
    maybeInliers := n randomly selected values from data
    maybeModel := model parameters fitted to maybeInliers
    alsoInliers := empty set
    for every point in data not in maybeInliers do
        if point fits maybeModel with an error smaller than t
             add point to alsoInliers
    end for
    if the number of elements in alsoInliers is > d then
        // This implies that we may have found a good model
        // now test how good it is.
        betterModel := model parameters fitted to all points in maybeInliers and alsoInliers
        thisErr := a measure of how well betterModel fits these points
        if thisErr < bestErr then
            bestFit := betterModel
            bestErr := thisErr
        end if
    end if
    increment iterations
end while

return bestFit
```

HOMOGRAPHY MATRIX – BRIEF OVERVIEW

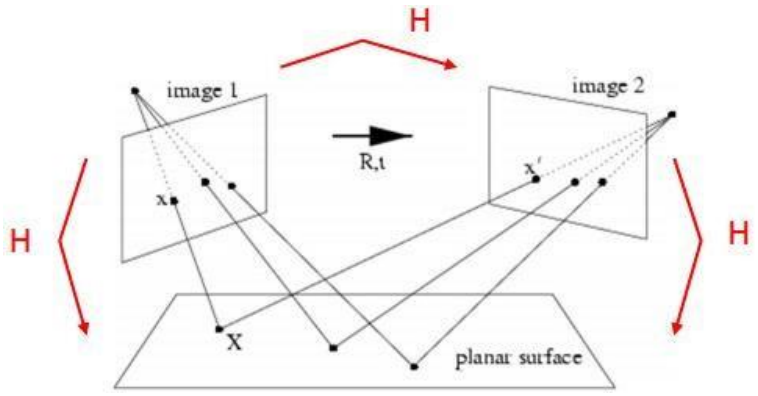
WHAT IS A HOMOGRAPHY MATRIX



$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Notes:

- A minimum of 4 points are needed to estimate the Homography (H) matrix
- Object must be on the same plan



OPENCV

In the following code, *pts_src* and *pts_dst* are numpy arrays of points in source and destination images. We need at least 4 corresponding points.

```
h, status = cv2.findHomography(pts_src, pts_dst)
```

The calculated homography can be used to warp the source image to destination. Size is the size (width,height) of im_ds

```
im_dst = cv2.warpPerspective(im_src, h, size)
```

Assignment 2 – SIFT Matching

Determine whether two images are similar to each other or not.

- You should use SIFT features to find the matches between each pair of images and filter these matches to get the most significant matches.
- For matches filtering; you have to use **at least three of these four techniques**:
  1- David Lowe's Ratio
  2- Threshold based on distance. (The distance must not be constant and should be derived from an analysis of the data distribution)
  3- Manual CrossCheck (Make sure that the top matches remain as the top if the order of query and train image changes). You cannot use built-in CrossCheck.
  4- Ransac
- For each pair of images calculate the similarity score using the equation below:

$$\frac{number\ of\ matches\ after\ filtering}{\min\left(number\ of\ keypoints\ in\ image1, number\ of\ keypoints\ in\ image2\right)}$$

- Draw the matches between the two images and print the similarity score between each two images and mention whether the two images are similar or not.
- No sharing of code or taking any portion of code from the internet is allowed. **Cheating detection will be applied.**
- **Assignment is Individual**
- **Assignment deadline: Friday 1/12/2023 11:59 PM**
- **Assignment Submission Form Link: [https://docs.google.com/forms/d/1iXWaFc-dV90HKaksKxheKfVVMyvCJs8OqjAqgqMFSu8/edit]**

Project Team Registration

- Team Registration Starts Friday 24/11/2023, Ends Tuesday 28/11/2023 11:59 PM.
- Each Team should consist of 4-6 members, Same Department.
- Top Three teams will be honored.
- Project Team Registration Link: [https://docs.google.com/forms/d/1mRMM6RqweYui6EOUehVBjTLtXy7TSpHrYh-fvQDH0yM/edit]

REFERENCE

[1] RANSAC: http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf

www.cse.psu.edu/~rtc12/CSE486/lecture15.pdf

[2] Interpretation of Lowe's ratio test: http://www.cs.technion.ac.il/~tammya/Publications/KaplanAvrahamLindenbaumECCV2016.pdf

[3] Homography: https://www.learnopencv.com/homography-examples-using-opencv-python-c/ https://docs.opencv.org/3.4.1/d9/dab/tutorial_homography.html